Strassen矩阵乘法

Strassen矩阵乘法

■传统方法：$C = A \times B$, C中的某个元素$C[i,j]$的计算公式为

$$C[i,j] = \sum_{k=1}^{n} A[i][k]B[k][j]$$

■时间复杂度：$O(n^3)$

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n×n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

Strassen矩阵乘法

- 分治法：将矩阵A，B和C分成4个大小相等的子矩阵。$C = A \times B$可写成：

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

由此可得：

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

- 时间复杂度： $O(n^3)$

$$T(n) = \begin{cases} O(1) & n = 2 \\ 8T(n/2) + O(n^2) & n > 2 \end{cases} \Rightarrow T(n) = O(n^3)$$

Strassen矩阵乘法

■Strassen矩阵乘法：为了降低时间复杂度，必须减少乘法次数(8→7)

$$M_1 = A_{11}(B_{12} - B_{22})$$

$$M_2 = (A_{11} + A_{12})B_{22}$$

$$M_3 = (A_{21} + A_{22})B_{11}$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_6 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_7 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$C_{11} = M_5 + M_4 - M_2 + M_6$$

$$C_{12} = M_1 + M_2$$

$$C_{21} = M_3 + M_4$$

$$C_{22} = M_5 + M_1 - M_3 - M_5$$

■时间复杂度： $O(n^{2.81})$

$$T(n) = \begin{cases} O(1) & n = 2 \\ 7T(n/2) + O(n^2) & n > 2 \end{cases}$$

$$T(n) = O(n^{\log 7}) = O(n^{2.81})$$

Strassen矩阵乘法

- 传统方法： $O(n^3)$
- Strassen矩阵乘法： $O(n^{2.81})$
- 更快的方法？
- Hopcroft和Kerr已经证明(1971)，计算2个2×2矩阵的乘积，7次乘法是必要的。因此，要想进一步改进矩阵乘法的时间复杂性，就不能再基于计算2×2矩阵的7次乘法这样的方法了。或许应当研究3×3或5×5矩阵的更好算法。
- 后来出现了其它算法，目前最好的时间上界为$O(n^{2.376})$

Strassen矩阵乘法：讨论

■Strassen矩阵乘法需要创建大量动态二维数组，分配堆内存空间会占用大量时间。

■改进：设定界限T，当矩阵规模小于T时，不再分治，而是使用传统方法进行计算。

动手题：在自己的计算机确定最佳的界限T'，即当矩阵规模小于T'时，传统方法所需的时间小于Strassen矩阵乘法